Remarks

Claim 1 is canceled herein. Claims 2, 6, 9, 14, 18 and 23 are amended herein. Claims 2-24 remain pending in the Application.

Claim Interpretation

In the Office Action, Claims 1, 9, 14, 18 and 23 are objected to because of informalities.

Applicants have canceled Claim 1 herein. Therefore, the objection with respect to Claim 1 is moot.

The Examiner has stated that it is unclear how the functional language "is dynamically tailored to the client to balance server-side and client-side execution and client-side storage requirements limits the claimed invention in compliance with 35 U.S.C. §112 , second paragraph.

Applicants have amended Claims 2, 9, 14, 18 and 23 to more clearly define the functional language. Therefore, Applicants respectfully submit that the claim interpretation objection is overcome.

Rejection under 35 U.S.C. §112

Claims 1-24

In the Office Action, Claims 1-24 are rejected under 35 U.S.C. §112, first paragraph, as failing to comply with the written description requirement.

Applicants have canceled Claim 1 herein. Therefore, the rejection with respect to Claim 1 is moot.

The Examiner has stated that the feature "wherein a granularity of a code segment is dynamically tailored to the client to balance server-side and client-

side execution and client-side storage requirements, and is configured based on predicted code segment usage or prior code segment usage history" was not described in the specification as originally filed. As such, the subject matter that provides adequate description and enabling support must be inserted.

Applicants respectfully disagree with the Examiner's rejection. Applicants respectfully submit that pages 12 -14 of the Specification in particular provide adequate description and enablement. Moreover, Applicants respectfully submit that the cited examples of the dynamic optimization scheme "dynamo" are not relied upon for enablement, but instead are cited as similar mechanisms for performing dynamic tailoring.

Applicants respectfully state that the specification including pages 12-14 clearly describe one method for dynamic optimization. Specifically, "a scheme called Dynamo. Basically, Dynamo begins execution by interpreting native binary code. While interpreting the code, Dynamo identifies "hot" code segments and identifies certain optimization opportunities. The "hot" code segments are then optimized and emitted into a segment cache, where the "hot" code segments are natively executed. Rarely executed code segments continue to be executed by interpretation. Accordingly, an application is executed by Dynamo using a mixture of native execution and interpretation. In the present invention, the techniques used to manage the code segments are similar, but are tailored to support execution in a client and server configuration."

Moreover, the Specification clearly defines a method for dynamic tailoring to include the granularity of the code segments can be easily adjusted to balance server-side and client-side execution and network bandwidth efficiency, and client-side storage requirements. Of course, the smallest possible code segment is a single binary instruction. However, many instructions do not branch, and execution continues with the next instruction in the sequence. Therefore, one

implementing the present invention would typically never select single instruction granularity. For all practical purposes, the smallest code segment size that one implementing the present invention would choose is a basic block of instructions. A basic block is a set of instructions bounded by either branch instructions or entry points. Accordingly, instructions in a basic block are always executed as a group starting with an entry point or an instruction after a first branch instruction and ending with an instruction associated with a second branch instruction. Basic block level granularity is still quite small. In typical binary code, on average there is a branch instruction every 5 or 6 instructions. Accordingly, one implementing the present invention may choose to use a larger code segment granularity. Alternatively, server code segment manager 22 of server 12 can start by using basic block granularity, and increase the code segment size as basic blocks that tend to be executed together are identified.

Applicants further submit that the complete Specification provides further details and enablement. For this reason, Applicants respectfully submit that the Claimed features are clearly supported, described and enabled in the Specification. As such, Applicants respectfully submit the rejection of Claims 2-24 under 35 U.S.C. §112 is overcome.

Rejection under 35 U.S.C. §103(a)

Claims 1-24

In the Office Action, the Examiner rejected Claims 1-20 under 35 U.S.C. §103(a) as being anticipated by Shimura (6370687). Applicants have reviewed the Office Action and believes the Examiner meant to reject Claims 1-24 under 35 U.S.C. §103(a) as being anticipated by Shimura and has responded accordingly. Applicants have reviewed Shimura and respectfully states that Shimura does not teach nor render obvious the present invention for the following rationale.

Applicants have canceled Claim 1 herein. Therefore, the rejection with respect to Claim 1 is moot.

Regarding Claim 2, Applicants respectfully submit that Claim 2 includes the features "an application code source that stores a client application; and a server code manager coupled to the application code source; an application code transformation manager coupled to the application code source, for transforming the client application from a first format to a native binary format compatible with a native instruction set of the CPU of the client."

Applicants respectfully submit that the claimed feature of the code being provided in native binary format to the CPU is not taught or rendered obvious by Shimura. That is, Applicants understand Shimura to teach virtual machine code operations. As clearly defined in the Summary portion of the Specification, this is significantly different than the native binary code supplied to the client in the present claimed feature. For this reason, Applicants respectfully submit that Shimura does not teach or render obvious the claimed features and as such Claim 2 overcomes the rejection under 35 U.S.C. §103(a).

Applicants respectfully submits that Claim 2 further includes the features "a server code segment manager coupled to the application code transformation manager, for parsing the client application in the native binary format into a plurality of code segments, wherein a granularity of the code segments is dynamically tailored to the client to balance server-side and client-side execution overhead and network bandwidth efficiency and client-side storage requirements, and is configured based on predicted code segment usage or prior code segment usage history that, wherein at least one of said plurality of code segments is are transmitted to the client via the network."

Applicants respectfully submit that the client application that has been transformed based on the native instructions of the client is then parsed into a plurality of code segments, wherein the parsing of the code segments, e.g., the granularity, is dynamically tailored to the client.

Applicants respectfully submit that Shimura does not teach this claimed feature. Applicants do not understand Shimura to teach dynamic parsing of application code into code segments. That is, assuming arguendo that Shimura parses application code into code segments. Applicants do not understand Shimura to teach parsing of code into code segments dynamically based on client and client server features. As stated, these features include execution overhead, bandwidth efficiency, storage requirements, and other features clearly described in the Specification. For this further reason, Applicants respectfully submit that Shimura does not teach or render obvious the claimed features and as such Claim 2 overcomes the rejection under 35 U.S.C. §103(a).

Applicants respectfully submits that Claim 2 also includes the features "a client coupled to the network said client not having said client application stored thereon, wherein the client comprises: a CPU for natively executing at least one of said plurality of said code segments derived from the client application stored on said server."

Applicants respectfully submits that Claim 2 also includes the features "a code cache coupled to the CPU, for storing said code segments, a client code manager coupled to the code cache, wherein the client code manager launches the client application by requesting that the server code manager transmit at least one of the plurality of code segments to the client, receiving at least one of the code segment from the server, storing the code segment in the code cache, and executing at least one of the plurality of code segments using the CPU until the executed code segment attempts to pass control to a required code segment

not stored in the code cache, at which point control passes back to the client code manager to retrieve the required code segment from the server, with the CPU continuing execution with the required code segment."

As the Examiner has stated on page 9 of the present Office Action, the Examiner cannot refute that Shimura does not teach storing the code segments on the client. The Examiner further states that the relevance is not understood.

Applicants respectfully state that the relevance of the claimed features is clearly defined in the specification including the Summary section and the last three paragraphs of page 25, which clearly state the relevance. Specifically, "the present invention is highly scalable since execution occurs on the client. In addition, since execution occurs on the client using native binary code segments stored in a code cache, the present invention is potentially much faster than prior art virtual machine (VM) interpreters. Since code segments can be stored and locked in the code cache of the client, execution can continue if the network connection to the server is temporarily interrupted. As discussed above, applications transmitted to the client are highly resistant to software piracy, making the present invention well suited for pay-per-use business models. In addition, since the application resides on the server, the user is relived of the burden of application management." For this additional reason, Applicants respectfully submit that the relevance is clearly defined. As such, Applicants respectfully submit, as the Examiner has stated, Shimura does not teach or render obvious the claimed features and as such Claim 2 overcomes the rejection under 35 U.S.C. §103(a).

Regarding Claim 9, Applicants respectfully submit that Claim 9 includes the features "wherein the server code manager derives native binary code segments in a native execution format required by client CPUs from the

application code source." Thus, the client application is transformed into a native binary format compatible with the native instructions of the Client.

Applicants respectfully submit that the claimed feature of the code being provided in native binary format to the CPU is not taught or rendered obvious by Shimura. That is, Applicants understand Shimura to teach virtual machine code operations. As clearly defined in the Summary portion of the Specification, this is significantly different than the native binary code supplied to the client in the present claimed feature. For this reason, Applicants respectfully submit that Shimura does not teach or render obvious the claimed features and as such Claim 9 overcomes the rejection under 35 U.S.C. §103(a).

Applicants respectfully submits that Claim 9 further includes the features "wherein a granularity of the code segments is dynamically tailored to the client to balance server-side and client-side execution overhead and network bandwidth efficiency and client-side storage requirements, and is configured based on predicted code segment usage or prior code segment usage history, and transmits at least one of the native binary code segments to said client not having said client application stored thereon upon requests received from said client."

Applicants respectfully submit that the client application that has been transformed based on the native instructions of the client is then parsed into a plurality of code segments, wherein the parsing of the code segments, e.g., the granularity, is dynamically tailored to the client.

Applicants respectfully submit that Shimura does not teach this claimed feature. Applicants do not understand Shimura to teach dynamic parsing of application code into code segments. That is, assuming arguendo that Shimura parses application code into code segments. Applicants do not understand

Shimura to teach parsing of code into code segments dynamically based on client and client server features. As stated, these features include execution overhead, bandwidth efficiency, storage requirements, and other features clearly described in the Specification. For this further reason, Applicants respectfully submit that Shimura does not teach or render obvious the claimed features and as such Claim 9 overcomes the rejection under 35 U.S.C. §103(a).

Regarding Claim 14, Applicants respectfully submit that Claim 14 includes the features "a CPU for natively executing code segments derived from the client application stored on the server."

Applicants respectfully submit that the claimed feature of the code being provided in native binary format to the CPU is not taught or rendered obvious by Shimura. That is, Applicants understand Shimura to teach virtual machine code operations. As clearly defined in the Summary portion of the Specification, this is significantly different than the native binary code supplied to the client in the present claimed feature. For this reason, Applicants respectfully submit that Shimura does not teach or render obvious the claimed features and as such Claim 14 overcomes the rejection under 35 U.S.C. §103(a).

Applicants respectfully submits that Claim 14 further includes the features "wherein a granularity of the code segments derived from the client application is dynamically tailored to the client to balance server-side and client-side execution overhead and network bandwidth efficiency and client-side storage requirements, and is configured based on predicted code segment usage or prior code segment usage history."

Applicants respectfully submit that the client application that has been transformed based on the native instructions of the client is then parsed into a

plurality of code segments, wherein the parsing of the code segments, e.g., the granularity, is dynamically tailored to the client.

Applicants respectfully submit that Shimura does not teach this claimed feature. Applicants do not understand Shimura to teach dynamic parsing of application code into code segments. That is, assuming arguendo that Shimura parses application code into code segments. Applicants do not understand Shimura to teach parsing of code into code segments dynamically based on client and client server features. As stated, these features include execution overhead, bandwidth efficiency, storage requirements, and other features clearly described in the Specification. For this further reason, Applicants respectfully submit that Shimura does not teach or render obvious the claimed features and as such Claim 14 overcomes the rejection under 35 U.S.C. §103(a).

Applicants respectfully submit that Claim 14 also includes the features "storing at least one of said code segments in the code cache, and executing at least one of said code segments using the client CPU."

As the Examiner has stated on page 9 of the present Office Action, the Examiner cannot refute that Shimura does not teach storing the code segments on the client. The Examiner further states that the relevance is not understood.

Applicants respectfully state that the relevance of the claimed features is clearly defined in the specification including the Summary section and the last three paragraphs of page 25, which clearly state the relevance. Specifically, "the present invention is highly scalable since execution occurs on the client. In addition, since execution occurs on the client using native binary code segments stored in a code cache, the present invention is potentially much faster than prior art virtual machine (VM) interpreters. Since code segments can be stored and locked in the code cache of the client, execution can continue if the network

connection to the server is temporarily interrupted. As discussed above, applications transmitted to the client are highly resistant to software piracy, making the present invention well suited for pay-per-use business models. In addition, since the application resides on the server, the user is relived of the burden of application management." For this additional reason, Applicants respectfully submit that the relevance is clearly defined. As such, Applicants respectfully submit, as the Examiner has stated, Shimura does not teach or render obvious the claimed features and as such Claim 14 overcomes the rejection under 35 U.S.C. §103(a).

Regarding Claim 18, Applicants respectfully submit that Claim 18 includes the features "deriving a plurality of code segments, in a native execution format required by the client, from an application code source stored on said server and not on said client."

Applicants respectfully submit that the claimed feature of the code being provided in native execution format is not taught or rendered obvious by Shimura. That is, Applicants understand Shimura to teach virtual machine code operations. As clearly defined in the Summary portion of the Specification, this is significantly different than the native binary code supplied to the client in the present claimed feature. For this reason, Applicants respectfully submit that Shimura does not teach or render obvious the claimed features and as such Claim 18 overcomes the rejection under 35 U.S.C. §103(a).

Applicants respectfully submit that Claim 18 further includes the features "wherein a granularity of the code segments is dynamically tailored to the client to balance server-side and client-side execution overhead and network bandwidth efficiency and client-side storage requirements, and is configured based on predicted code segment usage or prior code segment usage history."

Applicants respectfully submit that the client application that has been transformed based on the native instructions of the client is then parsed into a plurality of code segments, wherein the parsing of the code segments, e.g., the granularity, is dynamically tailored to the client.

Applicants respectfully submit that Shimura does not teach this claimed feature. Applicants do not understand Shimura to teach dynamic parsing of application code into code segments. That is, assuming arguendo that Shimura parses application code into code segments. Applicants do not understand Shimura to teach parsing of code into code segments dynamically based on client and client server features. As stated, these features include execution overhead, bandwidth efficiency, storage requirements, and other features clearly described in the Specification. For this further reason, Applicants respectfully submit that Shimura does not teach or render obvious the claimed features and as such Claim 18 overcomes the rejection under 35 U.S.C. §103(a).

Applicants respectfully submit that Claim 18 also includes the features "emitting at least one of the plurality of code segments into the code cache; and executing at least one of the plurality of code segments natively from the code cache."

As the Examiner has stated on page 9 of the present Office Action, the Examiner cannot refute that Shimura does not teach storing the code segments on the client. The Examiner further states that the relevance is not understood.

Applicants respectfully state that the relevance of the claimed features is clearly defined in the specification including the Summary section and the last three paragraphs of page 25, which clearly state the relevance. Specifically, "the present invention is highly scalable since execution occurs on the client. In addition, since execution occurs on the client using native binary code segments

stored in a code cache, the present invention is potentially much faster than prior art virtual machine (VM) interpreters. Since code segments can be stored and locked in the code cache of the client, execution can continue if the network connection to the server is temporarily interrupted. As discussed above, applications transmitted to the client are highly resistant to software piracy, making the present invention well suited for pay-per-use business models. In addition, since the application resides on the server, the user is relived of the burden of application management." For this additional reason, Applicants respectfully submit that the relevance is clearly defined. As such, Applicants respectfully submit, as the Examiner has stated, Shimura does not teach or render obvious the claimed features and as such Claim 18 overcomes the rejection under 35 U.S.C. §103(a).

Regarding Claim 23, Applicants respectfully submit that Claim 23 includes the features derive a plurality of code segments in a native execution format required by the client from an application code source stored on said server and not on said client."

Applicants respectfully submit that the claimed feature of the code being provided in native execution format is not taught or rendered obvious by Shimura. That is, Applicants understand Shimura to teach virtual machine code operations. As clearly defined in the Summary portion of the Specification, this is significantly different than the native binary code supplied to the client in the present claimed feature. For this reason, Applicants respectfully submit that Shimura does not teach or render obvious the claimed features and as such Claim 23 overcomes the rejection under 35 U.S.C. §103(a).

Applicants respectfully submit that Claim 23 further includes the features ", wherein a granularity of the plurality of code segments is dynamically tailored to the client to balance server-side and client-side execution overhead and network

bandwidth efficiency and client-side storage requirements, and wherein said granularity of said plurality of code segments is configured based on predicted code segment usage or prior code segment usage history, and transmit at least one of said plurality of said code segments to the client."

Applicants respectfully submit that the client application that has been transformed based on the native instructions of the client is then parsed into a plurality of code segments, wherein the parsing of the code segments, e.g., the granularity, is dynamically tailored to the client.

Applicants respectfully submit that Shimura does not teach this claimed feature. Applicants do not understand Shimura to teach dynamic parsing of application code into code segments. That is, assuming arguendo that Shimura parses application code into code segments. Applicants do not understand Shimura to teach parsing of code into code segments dynamically based on client and client server features. As stated, these features include execution overhead, bandwidth efficiency, storage requirements, and other features clearly described in the Specification. For this further reason, Applicants respectfully submit that Shimura does not teach or render obvious the claimed features and as such Claim 23 overcomes the rejection under 35 U.S.C. §103(a).

Applicants respectfully submit that Claim 23 also includes the features "third computer readable program code devices configured to cause the client to receive at least one of said plurality of said code segments, adjust branches in at least one of said plurality of said code segments having targets not in a code cache of the client to cause code segments containing the targets to be requested from the serve, emit at least one of said plurality of said code segments into the code cache, and execute at least one of said plurality of said code segments natively from the code cache."

As the Examiner has stated on page 9 of the present Office Action, the Examiner cannot refute that Shimura does not teach storing the code segments on the client. The Examiner further states that the relevance is not understood.

Applicants respectfully state that the relevance of the claimed features is clearly defined in the specification including the Summary section and the last three paragraphs of page 25, which clearly state the relevance. Specifically, "the present invention is highly scalable since execution occurs on the client. In addition, since execution occurs on the client using native binary code segments stored in a code cache, the present invention is potentially much faster than prior art virtual machine (VM) interpreters. Since code segments can be stored and locked in the code cache of the client, execution can continue if the network connection to the server is temporarily interrupted. As discussed above, applications transmitted to the client are highly resistant to software piracy, making the present invention well suited for pay-per-use business models. In addition, since the application resides on the server, the user is relived of the burden of application management." For this additional reason, Applicants respectfully submit that the relevance is clearly defined. As such, Applicants respectfully submit, as the Examiner has stated, Shimura does not teach or render obvious the claimed features and as such Claim 23 overcomes the rejection under 35 U.S.C. §103(a).

Therefore, Applicants respectfully submits that Shimura does not teach or render obvious the present claimed invention as recited in Claims 2, 9, 14, 18 and 23, and as such, Claims 2, 9, 14, 18 and 23 are in condition for allowance. Accordingly, Applicants also respectfully submits that Shimura does not teach or render obvious the present claimed invention as recited in Claims 3-8, 10-13, 15-17, 19-22 and 24 which are dependent on allowable Independent Claims 2, 9, 14, 18 and 23 and that Claims 3-8, 10-13, 15-17, 19-22 and 24 recite further features of the present claimed invention. Therefore, Applicants respectfully

states that Claims 3-8, 10-13, 15-17, 19-22 and 24 are allowable as pending from allowable base Claims.

Conclusion

In light of the above amendments and remarks, Applicants respectfully requests allowance of Claims 2-24.

The Examiner is invited to contact Applicants' undersigned representative if the Examiner believes such action would expedite resolution of the present application.

Respectfully submitted,
Wagner, Murabito & Hao LLP

Date: 6/2/06

John P. Wagner, Jr.
Reg. No. 35,398

Two North Market Street
Third Floor
San Jose, California 95113
(408) 938-9060